Hera-FFX: a Firefox add-on for semi-automatic web accessibility evaluation

José L. Fuertes
Dept. LSIIS
Technical University of Madrid
Campus de Montegancedo
28660-Boadilla del Monte
Madrid. Spain
+34 91 352 25 46
jfuertes@fi.upm.es

Ricardo González
Technical University of Madrid
Campus de Montegancedo
28660-Boadilla del Monte
Madrid. Spain
+34 91 352 25 46
rgonzalez@cettico.fi.upm.es

Emmanuelle Gutiérrez
Sidar Foundation
Sancho Dávila, 35
28028-Madrid. Spain
+34 91 725 71 47
emmanuelle@sidar.org

Loïc Martínez
Dept. LSIIS
Technical University of Madrid
Campus de Montegancedo
28660-Boadilla del Monte
Madrid. Spain
+34 91 352 25 46
loic@fi.upm.es

ABSTRACT
Website accessibility evaluation is a complex task requiring a combination of human expertise and software support. There are several online and offline tools to support the manual web accessibility evaluation process. However, they all have some weaknesses because none of them includes all the desired features. In this paper we present Hera-FFX, an add-on for the Firefox web browser that supports semi-automatic web accessibility evaluation.

Categories and Subject Descriptors
K.4.2 [**Computers and society**]: Social issues - *Assistive technologies for persons with disabilities*

General Terms
Human Factors

Keywords
Web accessibility, accessibility evaluation, evaluation tools

1. INTRODUCTION
It is a fact that web accessibility is gaining in importance in the international context, and especially in the European Union. In Europe, most countries have legislation stipulating that all public web sites have to be accessible in compliance with the World Wide Web Consortium's Web Content Accessibility Guidelines 1.0 (WCAG) [5].

In this context, the evaluation of website accessibility is of utmost importance. This evaluation cannot be fully automated, as many of the checkpoints require human judgment to assess a web page's conformity level. Thus, web accessibility evaluation is a complex task requiring human expertise and software support [6], [11]. The person performing this task needs sound knowledge and experience in web development and has to be proficient in the use of the techniques required to evaluate conformity for each of the WCAG 1.0 checkpoints.

Therefore, both expert and novice web accessibility evaluators have a common need: a tool that provides support for the manual evaluation of web accessibility, automating as much of the work as possible. We have previously presented such a tool, called Hera [1]. Hera is an online tool for semi-automatically evaluating website accessibility. Hera has been successfully used by partners of the Sidar Foundation [10] and the Technical University of Madrid to evaluate web sites and also as a supporting technology for teaching web accessibility [2].

Nevertheless, Hera has some limitations, which are what have motivated the work presented in this paper. The first weak point is that, being an online tool, Hera is not able to analyze local web pages unless the developer installs Hera locally (this means installing a restricted local web server with PHP and MySQL support, which then can access the local files.) The second drawback is related to the evaluation of web pages that require some sort of user authentication. Like almost all other existing tools, Hera often cannot analyze these restricted access pages. The third snag is that Hera is unable to evaluate the rendered version of a web page, including locally displayed styles and dynamically-generated content from scripts. Again this is a common limitation shared by most existing evaluation tools.

This paper presents Hera-FFX, an add-on for the Firefox web browser [7]. Hera-FFX overcomes the above difficulties by running an automatic preliminary evaluation of the web pages as they are browsed, as well as enabling the user to manually evaluate the accessibility of any of the pages.

The paper is structured as follows. Section 2 will briefly discuss desired features for a web accessibility evaluation tool. Section 3 will present an overview of related work, as a justification of the need for developing Hera-FFX. Section 4 will give a description of the main technical issues of the current version. Finally, section 5 will present some concluding remarks about user experiences with Hera-FFX, along with work to be undertaken in the future.

2. FEATURES OF WEB ACCESSIBILITY TOOLS
Several sources have outlined a set of relevant features for web accessibility evaluation tools. For instance, the W3C document on

W4A2009 - Technical, April 20-21, 2009, Madrid, Spain. Co-Located with the 18th International World Wide Web Conference.

selection of web accessibility tools [16], discusses accessibility, checkpoint coverage, configuration, integration, policy requirements, reliability, repair and web technology support.

Another relevant research on evaluating web accessibility tools is the one by Brajnik [3], which measures the tool effectiveness in fault identification and diagnosis using three basic concepts: completeness (how many accessibility defects present in the tested web site are caught by a tool and correctly shown to the user), correctness (proportion of problems reported by the tool that are indeed true problems) and specificity (the number of different possible issues that can be detected and described by a tool) of the tools.

Based on this previous work, on our day-to-day experience in the evaluation and teaching of web accessibility, and on the comments that we have received from the users of Hera (this tool has had several hundred users over the years, and some have suggested improvements that we have used in our requirements analysis), we can identify a list of relevant features for a web accessibility evaluation tool:

- *Automatic preliminary evaluation* (AE). Any tool should be able to automatically assess all the checkpoints (or parts of them) that can be automated. This is related to accessibility coverage (W3C) and completeness (Brajnik).
- *Support for manual filling* of checkpoints results (MF). Once the automatic evaluation is finished, the tool should provide automated support for the evaluator to fill-in the values of all the checkpoints (including the possibility of changing the values obtained in the automatic evaluation). Additionally, the user should be able to add comments about each checkpoint that could be used for later report generation. This is related to checkpoint coverage (W3C) and completeness and specificity (Brajnik).
- *Page presentation modification* for assisting checkpoint evaluation (PM). This modified presentation should highlight the elements that have to be inspected for a given checkpoint, and should display the relevant attributes of those elements. This is related to checkpoint coverage (W3C) and correctness (Brajnik).
- *Annotated code view* for assisting checkpoint evaluation (CV). The elements specified in the checkpoint should be highlighted in the HTML code. This is related to checkpoint coverage (W3C) and correctness (Brajnik).
- *Local pages evaluation* (LP). This feature is essential for web developers, as they should be able to assess the accessibility of web pages under development that are not published and without having to send the code to a remote server. This is related to configuration and integration (W3C), and completeness (Brajnik).
- *Restricted-access pages evaluation* (RA). The tool should be able to evaluate restricted access web pages (i.e. password-protected web sites) and secure pages (using the HTTPS protocol). This is related to configuration and integration (W3C), and completeness (Brajnik).
- *Rendered-page evaluation* (RP). The tool should be able to evaluate the rendered version of the page, which implies that it can evaluate locally displayed styles and dynamically-generated content from scripts. This is related to configuration, integration and web technology support (W3C), and completeness (Brajnik).
- *Report generation* (RG). The evaluators should be enabled to save reports based on the automatic and manual inspections in a handy format (that could be human or machine-readable). This is related to configuration and integration (W3C) and completeness, correctness and specificity (Brajnik).
- *Support for training* (ST). The tool should provide detailed information about each checkpoint, including normative text and techniques to be applied for assessment. This information is very useful for novice evaluators, as well as for persons that do not perform accessibility evaluations on a daily basis. This is related to checkpoint coverage (W3C) and completeness (Brajnik).
- *Multi-session capacity* (MS). Evaluating the accessibility of a web page is a long task, and can require several hours of time, depending on the complexity of the page under assessment. Typically, an evaluator will need to split the evaluation task into several work sessions. An evaluation tool should thus provide some multi-session capacity, enabling the user to store current work and to load this work later for resuming the assessment. This is related to configuration and integration (W3C) and to completeness (Brajnik).
- *Flexibility* to integrate other accessibility guidelines (FL). In addition to WCAG 1.0, there are other accessibility requirements for the web, such as the US 508 standards, the Spanish UNE 139803:2004 standard or the "*Barrierefreie Informationstechnik-Verordnung*" in Germany. In addition to that, the next version of WCAG is finished since December, 2008, and the tool should be flexible enough to easily incorporate the new success criteria. This is related to checkpoint coverage and policy requirements (W3C) and to completeness (Brajnik).

3. RELATED WORK AND MOTIVATION

Web accessibility evaluation tools are software programs or online services that help to determine whether a web site conforms to accessibility guidelines. While web accessibility evaluation tools can significantly reduce the time and effort it takes to evaluate web sites, no tool can automatically determine website accessibility [13]. There is a large number of existing evaluation tools in the domain of web accessibility, as can be found in [14]. In this section the focus is on tools listed by the W3C that are free of charge and that can evaluate the checkpoints of WCAG 1.0. We have decided not to include commercial tools in this comparison due to unfairness reasons. Firstly, some commercial tools are very expensive and thus unaffordable for many small- and medium-sized enterprises. Secondly, it is very difficult to gather precise information about the features of commercial tools, given the high testing costs.

These chosen tools are listed in alphabetical order in table 1, with their features.

Table 1. Summary of free web accessibility evaluation tools and their features

Tool	Type	AE	MF	PM	CV	LP	RA	RP	RG	ST	MS	FL
A-Checker	online	✓	✓	✓		upload				✓		
Accessibility Check	online	✓							✓			
EvalAccess	online	✓				copy HTML			✓			
Foxability	extension	✓				✓	✓	✓	✓			✓
Functional Accessibility Evaluator	online	✓							✓	✓		
Hera	online	✓	✓	✓	✓				✓ (+MR)	✓	✓	
HiSoftware® Cynthia Says™ Portal	online	✓							✓			
Mozilla/Firefox Accessibility Extension	extension	✓				✓	✓	✓	✓			
TAW Online	online	✓							✓			
TAW Standalone	application	✓	✓	✓	✓	✓	partial		✓ (+MR)		✓	✓
Torquemada	online	✓							✓			
Total validator	online	✓							✓			
WAVE	online	✓		✓		upload						
WAVE Firefox toolbar	extension	✓		✓		✓	✓	✓				
Web Accessibility inspector	application	✓				✓			✓			
Web Accessibility Self-Evaluation Tool	report		✓			✓	partial		✓	✓		

We also have decided not to include research-stage work on web accessibility evaluations. The efforts that we have found so far mainly focus on usability-based accessibility evaluation, which is extremely important during web site development. Our work, on the other hand, focuses on conformity assessment, that is, to check if a given web site conforms with a given set of requirements (in this case, the WCAG 1.0 checkpoints). It is recognized that user testing and other usability-based evaluations are useful for improving products but are not so useful for conformity assessment. And it has to be noted that in many countries web accessibility is mandatory (at least for public web sites). This implies that methods and tools are needed to evaluate web site legal accessibility (that is, the conformity with requirements).

The feature coverage of the tools listed in table 1 is summarized below:

- *Type of tool*: the vast majority are online services, although there are three browser extensions, two applications and one manual report.
- *Automatic evaluation* (AE): all the tools can perform automatic evaluation, with the exception of the last one, which consists only of a document for generating an accessibility report, along with online documentation for evaluating the checkpoints.
- *Manual filling* (MF): only four tools allow the user to manually fill-in the values for each checkpoint.
- *Page presentation modification* (PM): only five tools offer modified presentation of web pages for supporting manual evaluation.
- *Annotated code view* (CV): only two tools offer an annotated code view for supporting the evaluation of code-related checkpoints.
- *Local pages* (LP): more than 50% of the tools can evaluate local pages, although two require uploading files and one require copying HTML code. These two possibilities may not be convenient in cases where the local web pages contain sensible information.
- *Restricted-access pages* (RA): three tools can be used to evaluate any type of restricted-access pages (all of them are browser extensions) and two provide partial support to a limited number of situations (for example, TAW Standalone can be used to evaluate pages with require user identification, but it cannot be used for secure HTTP).
- *Rendered-page evaluation* (RP): only the three tools that are browser extensions can perform an evaluation of the web pages after script execution.
- *Report generation* (RG): almost all tools provide some type of report generation, although not all of them provide the same amount of detail. Two of the tools are able to generate reports in the EARL machine-readable language [15] (identified with "+MR").
- *Support for training* (ST): only four tools provide training-related content, such as information of detailed techniques to be used for evaluating each checkpoint.*Multi-session capacity* (MS): only two tools enable the user to store and recover current status of the evaluation of a web page.
- *Flexibility* (FL): only two tools provide extension mechanisms for adding new guidelines. Foxability uses JavaScript for the definition of new tests, whereas Taw Standalone uses regular expressions.

The conclusion of this analysis of related work is that there is no tool that covers all of the desirable features described in section 2. This has motivated our work. For each feature there is at least one tool covering it, but it is not practical for an evaluator having to rely on several tools to perform the accessibility evaluation task.

Thus, there is a need for a web accessibility evaluation tool that provides strong support for manual evaluation activities (like Hera and Taw Standalone), is able of evaluating all types of web pages (like the tools that are browser extensions) and includes an extension mechanism to incorporate new accessibility guidelines (like Foxability and Taw Standalone).

As we have been involved in the development of one of the listed tools, HERA [1], [8], [9], our approach has been to start with the features offered by HERA 2.0 and increment its capabilities. Briefly, the new tool should provide all the facilities that HERA 2.0 offers, plus the capability of evaluating any web page and the capability of being extended. To do this, the best solution is to develop a browser add-on, which we have called Hera-FFX and

which, initially, is being developed for Mozilla Firefox. In fact, Hera-FFX is a complete re-design and re-implementation of Hera.

4. HERA-FFX

4.1 Design

The main goals behind the design of Hera-FFX are: (1) to keep a similar level of usability of the one found in Hera 2.0 and (2) to be flexible enough so it can be easily extended with requirements and tests from other standards or recommendations.

Figure 1 shows a high-level diagram of Hera-FFX, underlying the main processes and which files or information structures are used. Below is a description of the most relevant elements:

- Hera-FFX stores the guideline definition and all the associated checkpoints and tests in a XML-based Configuration file. This file is loaded during the start-up of Firefox. Both the user interface of Hera-FFX and its internal behavior depend on the content on this XML file. This reduces the coupling between Hera-FFX and the accessibility requirements specification, increasing the flexibility of Hera-FFX. The definitions of the tests to be performed for each checkpoint are made in JavaScript, as is the case of the Foxability tool.

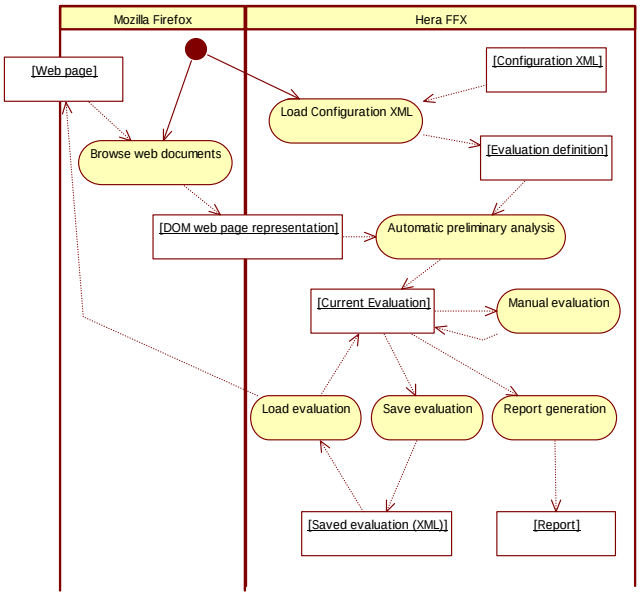

Figure 1. Hera-FFX global overview.

- The web pages are loaded and interpreted by Firefox, during the normal use of the browser. These pages can be either online or offline files, and can be unprotected or restricted pages. In fact, every page that can be displayed in Firefox can be analyzed by Hera-FFX.
- Once the web pages are interpreted by Firefox, a DOM representation of the web page is built, and this DOM representation is the one used by Hera-FFX to perform the accessibility evaluation.

The main tasks of Hera-FFX are to load the XML configuration file, to perform a preliminary automatic accessibility evaluation, to assist the user in a manual evaluation process, and to generate a report with the detailed results of the evaluation process. In addition, Hera-FFX can save and load the current evaluation results, enabling the user to divide his or her work into several sessions without trouble. These activities will be described in section 4.2.

The two main outputs of Hera-FFX are an accessibility report and the current evaluation stored in a XML file:

- *Accessibility report*. This report is generated when requested by the user. It is an XHTML+CSS accessible document containing detailed information of the current status of the accessibility evaluation activities. This detailed information includes the value given to each checkpoint of the accessibility guideline and, in addition, comments written by the evaluators during the assessment.
- *Saved evaluation* in a XML-based format. This evaluation can be loaded by Hera-FFX. This enables the user to split the evaluation of complex web pages into several working sessions.

During the evaluation process, Hera-FFX uses an internal structure that holds all the needed information: the guidelines, checkpoints and tests (read from the configuration file), the results of the tests (both automatic an manual), the DOM elements related to each test, the commentaries written by the evaluator, and so on. Below are the main elements of this internal structure:

- *Evaluation:* represents the current accessibility assessment. An evaluation contains general information, such as the URL of the page being evaluated and, in addition, points to each of the guidelines of the evaluation. The final result of an evaluation (the conformity level) is built from the result of each of the guidelines.
- *Guideline:* each of the high-level guidelines of one accessibility technical specification, as defined in the XML configuration file. One guideline contains general information about itself: the guideline text, instructions for the evaluator, identifier and, in addition, points to its checkpoints. The evaluation result of one guideline is built from the result of each of the checkpoints.
- *Checkpoint:* each of the detailed accessibility requirements belonging to one guideline, again as defined in the configuration file. Each checkpoint contains the checkpoint texts, instructions for the evaluator, detailed help and, in addition, points to a set of tests to be performed for evaluating the checkpoint. The result of one checkpoint is built from the results of each of its tests.
- *Test:* represents one of the tests to be performed during the accessibility evaluation. Some tests will be automatic (i.e., checking that each *<img>* element has an *alt* attribute), while other tests require manual evaluation (i.e., checking that the value of the *alt* attribute of an image is adequate). One test contains a textual description, the JavaScript code to be executed (if it is an automated test), explanations for the evaluator and, in addition, points to the elements in the page to be evaluated (this is relevant for the manual evaluation). The result of one test is built from the results of evaluating the test for each applicable element. In some cases there are tests without associated elements, because they are global to the page and require manual evaluation (one example is checking whether the presentation style is uniform across several pages of a web site).

- *Element:* represents one of the page elements associated with one test. Each element points to one DOM-Element (generated by Firefox) and adds useful information for the evaluation process: the more relevant is the automatic and manual result for the applied test. The Element class provides a separation layer between the tests and the actual DOM elements. It can happen that one DOM element is evaluated in several tests. In these situations the Element class stores the result of checking one test on one particular DOM element. In addition, the Element class contains information useful for showing both modified page presentations and annotated code views.
- *DOM-Element*: represents one of the DOM elements generated by Firefox during web browsing of the page under evaluation. This DOM element is unchanged by Hera-FFX and contains detailed information about all the attribute values for a given element. Hera-FFX thus uses the DOM to perform the accessibility evaluation, instead of the source code (as many other tools do). Using the DOM enables Hera-FFX to use the actual content of the web page when all the code is rendered by Firefox (including interpreting client-side scripting), instead of only relying on the source code contents. Thus Hera-FFX performs the evaluation of rendered pages, which are closer to the user experience.

The result assigned to each test during the automatic evaluation process can be one of the following:

- *Pass*: the web page passes the test; e.g. when the web page conforms to the markup language syntax (HTML, XHTML…).
- *Fail*: the web page fails the test; e.g. when an image has no *alt* attribute.
- *Verify*: the tool cannot decide what the result should be, and the user has not proceeded with the manual evaluation; e.g. to ensure that all information conveyed with color is also available without color.
- *N/A (Not applicable)*: the checkpoint is not applicable; e.g. the checkpoint is related to frames and the web site has none.

In addition, during the manual evaluation process, the user can assign two more values. These values are not assigned to individual tests, but to the whole checkpoint:

- *Partial*: the web page does not pass the checkpoint, but only for a minor reason (i.e., there are one hundred images and only one has an *alt* text that is not completely adequate).
- *Don't know*: the evaluator cannot decide the result for the checkpoint (for instance, if the user is blind and has to evaluate whether the alternative text of an image that he or she cannot see is adequate).

The final conformity result of one accessibility evaluation depends on the results obtained for each test. The result for each element of each test is propagated to the corresponding checkpoints, guidelines and, finally, the global evaluation.

The automatic propagation process from tests to checkpoints is as follows:

- If the result of one test for one element is "fail", then this result is assigned to the checkpoint independently of the results of the other tests associated with the checkpoint. However, these other tests are still applied and results are obtained, as this is relevant for a detailed manual evaluation and for generating the report.
- One checkpoint is assigned a "pass" value only if all of its tests have a "pass" or "not applicable" value.
- One checkpoint is assigned a "verify" value only if none of its tests have a "fail" value and there is at least one test with a "verify" value.
- One checkpoint will only have a "not applicable" value if all of its tests are "not applicable".

In addition, as it has been said above, the human evaluator can directly assign any value to any checkpoint, including the "partial" and "don't know" values.

From this description it can be deduced that Hera-FFX uses the following priority order of the values of tests and checkpoints:

Fail > Partial > Verify > Pass > Don't know > Not applicable

4.2 Detailed Evaluation Process

The Hera-FFX accessibility evaluation process is divided into three phases:

- *Automatic preliminary analysis*: Hera-FFX automatically analyzes the web pages that the Firefox user is browsing.
- *Manual evaluation support*: Hera-FFX offers the possibility of running a manual inspection to follow up the automatic assessment. This is a feature that few other accessibility evaluation tools offer.
- *Report generation*: Hera-FFX evaluators can save reports based on the automatic and manual inspections in a handy format.

Figure 3 shows each phase and the services offered to support manual inspection. As a manual evaluation of webpage accessibility is an extremely time-consuming process, Hera-FFX automatically runs a preliminary analysis that stops evaluators having to manually check each and every one of the 65 WCAG 1.0 checkpoints.

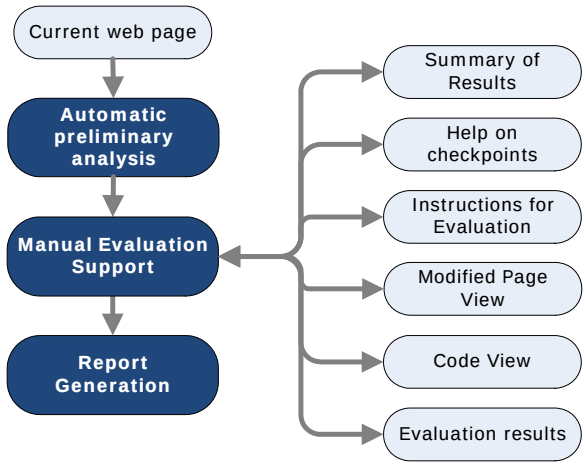

Figure 3. Overview of Hera-FFX evaluation process.

4.2.1 Automatic Preliminary Analysis

Once the browser completely loads a page, Hera-FFX runs an automatic analysis. One reason for waiting for the page to load

completely is because many web pages are likely to contain dynamic elements that are added at load time. These dynamic elements can be generated by JavaScript, server programs (PHP, JSP, etc.) or any rendering technology that modifies the browser's DOM representation with respect the web site's original HTML source code.

Even restricted-access pages (either password-protected or pages with secure access through HTTPS) can be analyzed by Hera-FFX. In fact, any web page that Firefox is able to render, with the corresponding DOM model created, can be analyzed by Hera-FFX. As shown in table 1, this feature is only offered by four other tools.

Once the page has been fully loaded (with dynamic elements and the browser's DOM representation), the tool runs a user-transparent automatic accessibility analysis. During this automatic process each DOM element is evaluated for conformity with the relevant automatic tests. The assigned values and the relationship between tests and DOM elements are stored in the internal information structure, as was described above.

Whilst the automatic analysis process is in progress, the Hera-Ext icon appearing in Firefox's status bar changes into an hourglass, indicating that the test is running. At the end of the test, the hourglass is replaced by a result icon that indicates whether the visited page contains any fail or there are only checkpoints to be checked manually. This way, users are informed at all times about what the tool is doing and the result.

The fact that the tool is embedded in the browser speeds up the automated and controlled checking process and test results generation. The user can stop the tool at any time, switching to idle mode.

Apart from the details provided by the Hera icon, users can get more detailed tabulated information by simply positioning the mouse cursor on the icon (Figure 4). This table represents the number of points to be checked, as well as incorrect, correct and not applicable points arranged by priorities 1, 2 and 3. Like the icon, the table is dynamic. Therefore, the values in the table cells will change, depending on the result of the automatic analysis, every time a new web page is visited or reloaded.

Figure 4. Hera-FFX overview of automatic results.

The Hera-FFX icon state could possibly change after the results have been generated. This is because HTML and CSS syntax validation services are used through AJAX technology (Asynchronous JavaScript and XML). Because the response is asynchronous, it could change the Hera-FFX icon.

This preliminary analysis automatically assigns a result to each of the 65 checkpoints. This result can only be: pass, fail, not applicable or verify. Table 2 shows a summary of the checkpoints that Hera-FFX can evaluate automatically.

Table 2. Checkpoints automatically analyzed by Hera-FFX

	Verify	*Pass*	*Fail*	*N/A*
Priority 1	9	-	3	5
Priority 2	16	1	7	4
Priority 3	10	2	2	4

Several checkpoints appear in more than one column in Table 2. For instance, checkpoint 1.1 (text alternatives) could be automatically evaluated as "fail" if there are images without the *alt* attribute, as "not applicable" if the page has no non-textual elements (images, objects, etc.), and as "verify" if there are images with *alt* attributes that require human evaluation to assess whether the alternative text matches the image. On the other hand, this checkpoint will never be automatically evaluated as "pass", because the computer cannot judge the adequacy of alternative texts.

4.2.2 Summary of Results

The summary of the test results is an on-demand functionality that the tool offers. This functionality is accessed by double clicking on the Hera-FFX icon in the browser status bar or by activating an equivalent command in the tool's menu. This summary of results is shown in a new window (which is a Firefox user interface, not browser window), displaying data like:

- *URL*: web address of the page under analysis.
- *Date/time*: date and time of the automated inspection.
- *Total*: number of elements in the page.
- *Automated analysis*: time (in seconds) taken by the automatic process.
- *Errors*: number of failed checkpoints detected during the automated analysis.
- *For manual checking*: number of checkpoints that HERA-FFX has determined should be evaluated by the user.
- *Browser*: name and version of the user's browser and the operating system.

Apart from these data, users can decide on the navigation style they want to use to get detailed automatic results and then conduct a manual inspection supported by the Hera-FFX tool:

- *Table-based navigation style*, where each table cell represents the test checkpoints grouped by results and associated priority. This navigation style is useful for locating what checkpoints the web page fails to conform to and offers several scanning strategies. For instance, some evaluators choose to follow the priority levels, whereas others prefer to focus first on the failed checkpoints (irrespective of their priority level), then on the checkpoints requiring manual inspection to finish with the passed and not applicable checkpoints. Note that, although the tool performs a preliminary evaluation and decides whether some checkpoints pass, fail or are not applicable, the responsibility ultimately falls to the human evaluator.
- *Guideline-based navigation style*, displaying one button for every guideline. This navigation style follows the WCAG 1.0 order, i.e. the numbering of the checkpoints as defined in the 1999 recommendation. Some users feel more at home following the WCAG reading order, as this is the way in

which they are accustomed to reading and understanding the checkpoints.

Having decided on the navigation style (by clicking on a table cell for the point-by-point table-based navigation style or a guideline button for guideline-based navigation), the tool offers rapid, simple and effective access to the results, as shown in Figure 5.

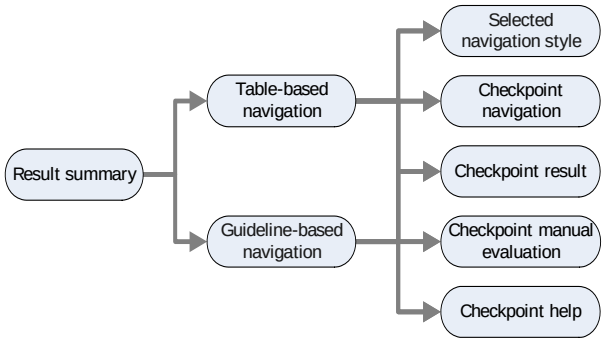

Figure 5. Overview of Hera-FFX navigation.

Users selecting a specific checkpoint are provided with detailed help explaining the objective and how to inspect the point, a manual inspection window and help for modifying the automatic evaluation run by the tool and a visualization of the results for the checkpoint. Figure 6 shows a screenshot of the user interface of Hera-FFX when using the table-based navigation style. The tool is listing all priority 2 checkpoints and the user has selected checkpoint 3.3, which currently has a "fail" value automatically assigned. In the lower right corner the tool shows the interactive interface to modify this value and to write a comment.

Figure 6. Hera-FFX user interface.

The following design points were taken into account:

- Interface control, as the interface is both keyboard and mouse operated.
- The color combination used should respect the contrast values dictated by the WCAG.
- Colors provide additional information (content reinforcement) and are accompanied by an icon and text to represent the test results. For example, a red X-shaped cross on a light red background represents a failure to comply with a checkpoint. This makes the results analysis more understandable.
- All the interface elements, like text and images, are specified in relative units and fit to screen size no matter what hardware is used.

4.2.3 Manual Evaluation Support

Hera-FFX can be used to manually inspect each checkpoint, irrespective of the chosen navigation style.

Because of the tremendous amount of information that has to be dealt with during a manual inspection, the user interface is designed so that evaluators can carry out the evaluation point-by-point according to the navigation style selected at the start. Other accessibility tools fail to provide such easy navigation and quick access to results. This slows down the job of inspection enormously.

The intermediate internal structure used in the automatic phase is also used during the manual phase to perform each of the Hera-FFX functionalities, as described below

To help to give a better understanding of the checkpoints and ease manual inspection, the following results visualization services are provided for each of the checks run to verify a specific point:

- *Simulated view* of the modified page indicating which elements have to be inspected (figure 7). As shown in table 1, very few existing tools offer this feature: only A-checked, Hera and Taw Standalone. What Hera-FFX has to do to highlight the relevant elements is to compare the stored elements in the internal structure with the actual elements in the web page. Once the elements are found, Hera-FFX transforms them in several ways (like adding an icon, a border and a background color) so that the elements are easy to be found by the human evaluator. This is a good option to use when the elements that have need of manual inspection are visible in the page, such as images, whose alternative text can be shown in the simulated view. In no case is the original web page being tested (corresponding to the Firefox browser window) altered. The elements specified in the results for the checkpoint are highlighted on a colored background together with an icon that varies depending on the result: blue and magnifying glass if it is to be verified, red and cross icon if it is fail, or green and check icon if it is correct.
- HTML *code view* generated by the page (remember that the generated HTML code will not necessarily be exactly the same as the web site source code, primarily because of dynamic elements generated at browser page load time). Only two existing tools offer this feature (Hera and Taw Standalone). As in the simulation view, the elements specified in the checkpoint results are highlighted on a colored background accompanied by an icon that varies depending on the result (figure 8). This view is a good option, for example, when the web site contains elements that are not easily visible, like, for example, JavaScript code, language changes, etc.
- *Visualize the result of a call to an external service*. This is a checkpoint-specific visualization and refers to a response by an external web service like, for example, HTML and CSS syntax validation. Internally, Hera-FFX uses AJAX technology to call to the respective service. This approach then depends on the offered service (e.g. on the W3C

validation service in the case of validation), which it displays as an HTML document in another application tab.

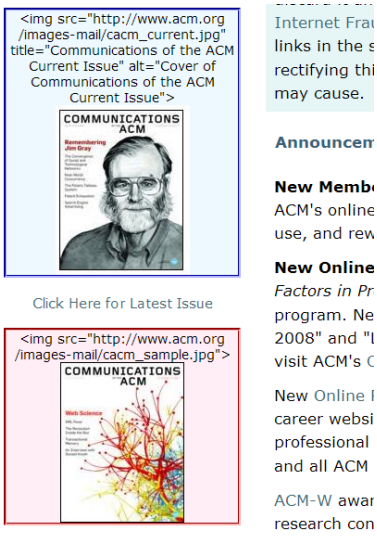

Figure 7. Example of simulated page view

Figure 8. Example of annotated code view

When inspecting a checkpoint, it should either be classed as pass, fail, partially incorrect, not applicable, or identified as don't know. Additionally, comments about the checkpoint can be added and stored for later report generation. All this information is also stored in the internal structure, in such a way that the manual results annotated by the user always prevail over the automatic results.

To further ease the efficiency of the process, the changes made by the evaluator are automatically accepted when the user moves to another checkpoint (without need to press any "save" or "apply changes" button). When this happens, Hera-FFX performs some changes in the user interface:

- Result visualization in the "summary" tab. When there is a change in the result of one checkpoint, the corresponding cells in the table are updated (typically there is a cell that will increment its count and other that will decrement it). For instance, if one checkpoint on priority 2 passes from "pass" to "partial", the second row cell under "pass" will decrement its count and the cell under "partial" will increment it.
- Result visualization in the reduced table of the "tests" tab: changes in the numbers will happen in a similar way as for the summary table.
- Checkpoint result visualization in the list of checkpoints. The text, color and icon of the checkpoint will change accordingly with the new value for that checkpoint.
- Checkpoint result visualization in the detailed view of checkpoints. Again, the text, color and icon will be updated.

A checkpoint will be in one or other table cell depending on the result returned in the preliminary analysis.

Finally it has to be noted that each time a page is reloaded in Firefox, the internal structure is deleted and rebuilt in order to store the data of the new analysis of the web page.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a Firefox add-on that performs a semi-automatic evaluation of the accessibility of the web sites that are being browsed. Hera-FFX first performs an automatic preliminary evaluation and then enables the user to analyze the detailed results and to proceed with the manual evaluation of all 65 WCAG 1.0 checkpoints.

Hera-FFX has several advantages over other existing tools. It includes in one package all the desirable features in a web accessibility evaluation tool (stated in section 2). Table 1 showed that no tool included all these features at the same time, so Hera-FFX fills an existing gap in this area, as Table 3 below summarizes (the value column includes a "new" label if the feature was not present in Hera online).

We feel that Hera-FFX's combination of features is a significant contribution. The main difference is that Hera and Hera-FFX are focused on the manual evaluation of web accessibility, which is not the case of the other tools. In our experience this greatly affects the efficiency and effectiveness of performing manual evaluations of web accessibility

In addition, Hera-FFX improves the Hera online tool providing few new characteristics: First, it can evaluate local files. Second, it can evaluate restricted access pages. Third, it can evaluate the content rendered after the web browser has processed all the dynamic content. And finally, it improves flexibility as it is possible to modify or add checkpoints and tests.

One key point that accessibility evaluators should take into account is the results of evaluating a web page with Hera-FFX (or other tools implemented as browser extensions) comparing with the results of evaluating the same web page with Hera (or other online or standalone tools). Extensions tools evaluate the web page after it has been rendered by the browser, having executed its scripts, so they evaluate the rendered code, whereas the other tools evaluate just the source code. This could lead to different values to some of the checkpoints.

Table 3. Summary of Hera-FFX features

Feature	Value	Comment
Automatic preliminary evaluation	✓	The "Atomatic preliminary analysis" Hera-FFX module automatically evaluates the checkpoints once the browser had completely loaded a web page and the all the dynamic elements had been generated. It provides a preliminary result for the checkpoints.
Support for manual filling	✓	The user is guided to complete the automatic preliminary evaluation with a manual evaluation of the web page. Hera-FFX provides support offering different services to perform the manual evaluation and forms to add results and comments.
Page presentation modification	✓	Elements to be manually evaluated are highlighted in a modified page view. This allows user to easily locate and check the conformity of checkpoints.
Annotated code view	✓	Elements to be manually evaluated are highlighted in a code view. This allows user to quickly locate the lines of code with interesting labels or attributes to check the conformity of checkpoints.
Local pages evaluation	✓ new	Hera-FFX is able to work with local web pages. This is possible because the evaluation is performed after the web page is loaded in the browser, so no matter where the page was obtained.
Restricted-access pages evaluation	✓ new	Hera-FFX is able to work with restricted web pages (HTTPS, password-protected…). This is possible because the evaluation is performed after the web page is loaded in the browser and all restrictions have been surpassed by the user.
Rendered-page evaluation	✓ new	Hera-FFX performs the evaluation when the web page is fully loaded. This means that all the dynamic content has been processed and generated.
Report generation	✓	The "Report generation" module generates a full HTML report about the accessibility of a web page. This report includes both the automatic and manual evaluation results.
Support for training	✓	The "help on checkpoints" and the "instructions for evaluation" services provide complementary information to the evaluator. This helps him or her to better understand the checkpoint under evaluation and to learn how to check it for comformity.
Multi-session capacity	✓	The "Load evaluation" and "Save evaluation" modules allows to store the ongoing evaluation. This can be used to interrupt the evaluation or to interchange evaluations among different evaluators.
Flexibility	✓ new	All the structure of guidelines and checkpoints are stored in XML files; together with JavaScript code that implements the automatic evaluation techniques. This allows flexibility as users could change both the guidelines and checkpoints as well as the evaluation code.

These advantages have been built into a user interface that has been designed to closely emulate the Hera online tool's behavior and, this way, facilitate the transition from the online tool (which has been considered to be extremely useful by its hundreds of users) to Hera-FFX. This was demonstrated by a preliminary user evaluation involving 6 expert users of Hera. It is an informal usability evaluation focused on making sure that the tool is useful for people that are proficient with Hera. These 6 people had to evaluate several web pages using Hera and Hera-FFX. Each person only used one tool for a given web page (i.e. for a given page, 3 people used Hera and 3 other people used Hera-FFX). We obtained the following usability results:

- Increased efficiency. One of the main reasons of this better efficiency is that each manual evaluation is automatically updated in Hera-FFX. In the Hera tool, users were obliged to click on the "record results" button to store the results of manual evaluation.
- Same effectiveness. No relevant changes on effectiveness were measured.
- Increased user satisfaction. The main reason is because users now are able to evaluate web pages that were impossible with Hera online (rendered, local or restricted pages).

This is a preliminary evaluation with a very limited number of users. For this reason, we think that the results are not conclusive and we plan to carry out a more detailed usability evaluation to assess how usable Hera-FFX is in different contexts of use.

With the same users we also did a preliminary evaluation of the tool effectiveness, according to Brajnik's proposal. When compared to Hera-online, we obtained the following results:

- Same completeness. Both tools allow the user to fully evaluate the 65 checkpoints of WCAG 1.0.
- Slightly increased correctness, due to the fact that Hera-FFX inspects the rendered-page, and is closer to the user experience.
- Same specificity. Both tools offer the same coverage of WCAG.

Concerning future work, we are currently working on extending the Hera-FFX functionality by generating reports in different formats (PDF and EARL). Although the add-on is in Spanish, it has been designed to be easily internationalized. We plan to have an English version ready in several months' time.

In addition we also plan to update Hera-FFX to cover the next version of WCAG, which was published as a final recommendation on December 11, 2008 [4], and to cover the Spanish web content accessibility standard [12].

Concerning WCAG 2.0, the work of adapting Hera-FFX has already started. WCAG 2.0 has a more complex structure with principles, guidelines, success criteria, techniques (both sufficient and advisory) and failures. Our plan is to consider techniques and failures at the same level as current "tests" in Hera-FFX, but some structural changes are needed to handle the flexibility in WCAG

2.0 to choose between multiple techniques to satisfy a requirement.

In the longer term, our ultimate goal is to develop a complete stand-alone evaluation tool including workgroup tools, such as project creation and management, evaluation work distribution, aggregation of results from different evaluators, assessment of evaluator reliability and so on.

6. ACKNOWLEDGMENTS

The authors wish to thank the Sidar Foundation and especially Carlos Benavidez for developing Hera, the tool that our add-on is based on.

The work described in this paper was funded in part by the Technical University of Madrid and Madrid's Regional Government as part of support received for "Assistive Computer Technology for Disabled People" (CCG06-UPM/INF-388).

7. REFERENCES

[1] Benavídez, C., Fuertes, J. L., Gutiérrez, E., and Martínez, L. 2006. Semi-Automatic Evaluation of Web Accessibility with HERA 2.0. Lecture Notes in Computer Science 4061 (July 2006), 199-206.

[2] Benavídez, C.; Fuertes, J. L.; Gutiérrez, E.; and Martínez, L. 2006. Teaching Web Accessibility with "Contramano" and Hera. Lecture Notes in Computer Science 4061 (July 2006), 341-348.

[3] Brajnik, G. 2004. Comparing accessibility evaluation tools: a method for tool effectiveness. Universal Access in the Information Society. 3(3-4), Springer Verlag, pp. 252-263.

[4] Caldwell, B., Cooper, M., Reid, L. G., and Vanderheiden, G. (eds). 2008. Web Content Accessibility Guidelines 2.0. World Wide Web Consortium Recommendation (December 2008). http://www.w3.org/TR/WCAG20/.

[5] Chisholm, W., Vanderheiden, G., and Jacobs, I. (eds.) Web Content Accessibility Guidelines 1.0. World Wide Web Consortium Recommendation (May 1999). http://www.w3.org/TR/WCAG10.

[6] Education and Outreach Working Group. 2006. Evaluating Web Sites for Accessibility: Overview. World Wide Web Consortium. http://www.w3.org/WAI/eval.

[7] Mozilla. 2008. Firefox web browser. http://www.mozilla.com/en-US/firefox/.

[8] Sidar Foundation. 2003. Hera tool Version 1.0 http://www.sidar.org/ex_hera/ index.php.en.

[9] Sidar Foundation. 2005. Hera 2.0: Accessibility testing with style. URL= http://www.sidar.org/hera/index.php.en.

[10] Sidar Foundation. 2008. Fundación Sidar - Acceso Universal, Seminario SIDAR (in Spanish). http://www.sidar.org/.

[11] Slatin, J. M. and Rush, S. 2003. Maximum Accessibility: Making Your Web Site More Usable for Everyone. Addison Wesley Professional, Boston.

[12] Spanish Association for Standardization. 2004. Computer applications for people with disabilities. Web content accessibility requirements. Spanish standard UNE 139803:2004. Madrid: AENOR (in Spanish).

[13] W3C. 2006. Web Accessibility Evaluation Tools: Overview. Web Accessibility Initiative, World Wide Web Consortium. http://www.w3.org/WAI/ER/tools/.

[14] W3C. 2008. Complete List of Web Accessibility Evaluation Tools. http://www.w3.org/WAI/ER/tools/complete

[15] W3C. 2008. EARL Overview. http://www.w3.org/WAI/intro/earl.php

[16] W3C. 2008. Selecting Web Accessibility Evaluation Tools. Draft. http://www.w3.org/WAI/eval/selectingtools.html